**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Методы обработки и анализа медицинских изображений»**

**Тема: СИНТЕЗ ПАНОРАМНЫХ ИЗОБРАЖЕНИЙ**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 8308 |  | Сергеев В.С. |
|  |  | Петров Г.А. |
|  |  | Лукашов Н.С. |
| Преподаватель |  | Поздеев А.А. |

Санкт-Петербург

2023

**Цель работы**

Изучение методов синтеза панорамы и их практическая реализация.

**Теоретические основания**

Общепринятый алгоритм для синтеза панорам состоит из следующих шагов.

1. Нахождение характерных точек (в англоязычной литературе используют термин «ключевая» точка «key point») на исходных изображениях в зоне перекрытия. Под характерной точкой понимают некоторый малый фрагмент изображения, в котором как значение яркостного градиента, так и производная (скорость изменения) градиента по направлению высоки. Как правило, характерных точками являются различные углы на изображениях (рис. 3.1).

2. Нахождение в зоне перекрытия одних и тех же особенностей на различных снимках (рисунок 3.2). То есть определение соответствующих друг другу пар характерных точек

Согласованные пары точек — это основа для синтеза панорамы. С помощью установленных пар на последующих этапах решают задачи калибровки, трансформации и объединения снимков. В случае синтеза панорамы из нескольких кадров (частей) при неизвестном заранее расположении каждого снимка, на данном этапе проводят процедуру регистрации – идентификации местоположения отдельных изображений на общей панораме и установление взаимных соответствий характерных точек.

3. В современных программных пакетах следующим шагом алгоритма сшивки является калибровка изображений. Эта процедура направлена на минимизацию искажений объектива, оптических дефектов, различий экспозиции. С помощью информации о согласованных парах характерных точек минимизируют влияние дисторсии (геометрических искажений) объектива на точность сшивки панорамы.

4. Ключевым этапом является процедура идентификации параметров уравнений трансформации изображений и последующее преобразование фрагментов с объединением в единую панораму. Данный этап требует задания вида трансформации, которое определяется типом создаваемой панорамы. Например, при отсутствии, или несущественности перспективных искажений у фрагментов имеет смысл использовать аффинное преобразование. Примером такого изображения может быть «сшивка» панорамы из отсканированных частей единого документа (географической карты, картины, и т.п.). В общем случае применяют перспективное преобразование, учитывающее все возможные искажения снимков.

5. Заключительным этапом является блэндинг. Это комплексная процедура, направленная на повышение визуального качества панорамы, включающая выравнивание яркости и цветовой палитры фрагментов, маскирование «швов», удаление «призраков» (движущихся объектов). Кроме того, в блэндинг включают и процедуру проецирования панорамы на заданную поверхность – сферическую, цилиндрическую, эквидистантную и пр.

**Задание**

Разработать программу, реализующую синтез панорамы из двух фрагментов с помощью библиотеки OpenCV.

Необходимо подобрать два фрагмента (зона перекрытия изображений должна занимать не менее 25% их площади, желателен поворот или перспективное преобразование одного изображения относительно другого) осуществить синтез панорамы.

При работе программы должны быть визуализированы основные этапы:

* + изображения с найденными ключевыми точками;
  + результат процедуры поиска согласованных пар;
  + результат трансформации изображения (один из снимков является опорным и остается без изменений, трансформируют только второй фрагмент);
  + результат построения панорамы.

**Ход работы**

Изображения с найденными ключевыми точками рисунках 1 и 2 соответственно.

****

Рисунок 1 – Левое изображение с ключевыми точками

****

Рисунок 2 – Правое изображение с ключевыми точками

Результаты процедуры поиска согласованных пар представлен на рисунке 3.



Рисунок 3 – Результат процедуры поиска согласованных пар

Результат трансформации правого изображения представлен на рисунке 4.

****

Рисунок 4 – Результат трансформации правого изображения

Результат построения панорамы представлен на рисунке 5.



Рисунок 5 – Полученная панорама

**Вывод**

В ходе выполнения лабораторной работы были изучены методы синтеза панорамы, а также реализованы на практике с визуализацией результатов на каждом шаге общего алгоритма.

Приложение А. Листинг исходного кода.

#include "opencv2/core/core.hpp"

#include "opencv2/features2d/features2d.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/calib3d/calib3d.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/stitching.hpp"

#include "opencv2/core/cvstd.hpp"

#include <iostream>

#include "opencv2/core.hpp"

//#include "opencv2/xfeatures2d.hpp"

#include <math.h>

#include <iostream>

**using** **namespace** cv;

**using** **namespace** std;

**int** **main**()

{

**int** colorType = **1**;

Mat img1 = imread("i3.jpg", colorType);

**if** (img1.empty()) {

**return** **0**;

}

Mat img2 = imread("i4.jpg", colorType);

**if** (img2.empty()) {

**return** **0**;

}

//Инициализация детектора, подготовка контейнеров для характерных точек и их дескрипторов

Ptr<ORB> detector = ORB::create();

std::vector<KeyPoint> keypoints1, keypoints2;

Mat descriptor1, descriptor2;

//Поиск характерных точек на сшиваемых изображениях и вычисление их дескрипторов

detector->detect(img1, keypoints1);

detector->detect(img2, keypoints2);

detector->compute(img1, keypoints1, descriptor1);

detector->compute(img2, keypoints2, descriptor2);

//Отрисовка найденных характерных точек

Mat keypoints1draw, keypoints2draw;

drawKeypoints(img1, keypoints1, keypoints1draw);

drawKeypoints(img2, keypoints2, keypoints2draw);

imshow("keypoints1draw", keypoints1draw);

imshow("keypoints2draw", keypoints2draw);

//Сопоставление дескрипторов точек с помощью BFMatcher

BFMatcher matcher(NORM\_HAMMING);

std::vector<std::vector<DMatch>> matches;

matcher.knnMatch(descriptor1, descriptor2, matches, **2**);

std::vector<KeyPoint> matched1, matched2;

std::vector<DMatch> good\_matches;

**float** match\_ratio = **0.5f**;

**for** (**size\_t** i = **0**; i < matches.size(); i++) {

DMatch first = matches[i][**0**];

**float** dist1 = matches[i][**0**].distance;

**float** dist2 = matches[i][**1**].distance;

**if** (dist1 < match\_ratio \* dist2) {

**int** new\_i = **static\_cast**<**int**>(matched1.size());

matched1.push\_back(keypoints1[first.queryIdx]);

matched2.push\_back(keypoints2[first.trainIdx]);

good\_matches.push\_back(DMatch(new\_i, new\_i, **0**));

}

}

Mat dMatches;

drawMatches(img1, matched1, img2, matched2, good\_matches, dMatches);

imshow("Matches", dMatches);

//--------------------------------------------------------------------------------------------------------------------------

// Оценка матрицы гомографии

std::vector<Point2f> matched1\_pts, matched2\_pts;

**float** minX = img1.size().width;

**for** (**int** i = **0**; i < good\_matches.size(); i++) {

**if** (minX > matched1[good\_matches[i].queryIdx].pt.x)

minX = matched1[good\_matches[i].queryIdx].pt.x;

matched1\_pts.push\_back(matched1[good\_matches[i].queryIdx].pt);

matched2\_pts.push\_back(matched2[good\_matches[i].trainIdx].pt);

}

Mat H = findHomography(matched1\_pts, matched2\_pts, RANSAC);

Mat alignedImg2 = Mat::zeros(img2.size()\***2**, img2.type());;

warpPerspective(img2, alignedImg2, H.inv(), alignedImg2.size());

imshow("alignedImg2", alignedImg2);

Mat final = Mat::zeros(alignedImg2.size(), alignedImg2.type());

Mat roi1(final, Rect(**0**, **0**, alignedImg2.cols, alignedImg2.rows));

alignedImg2.copyTo(roi1);

Mat roi2(final, Rect(**0**, **0**, img1.cols, img1.rows));

img1.copyTo(roi2);

imshow("final", final);

imwrite("/keypoints1draw.jpg", keypoints1draw);

imwrite("/keypoints2draw.jpg", keypoints2draw);

imwrite("/dMatches.jpg", dMatches);

imwrite("/alignedImg2.jpg", alignedImg2);

imwrite("/final.jpg", final);

waitKey();

**return** **0**;

}